A method for the acquisition of ontology-based user profiles

Lixin Han\textsuperscript{a,b,c,*}, Guihai Chen\textsuperscript{b}, Ming Li\textsuperscript{d}

\textsuperscript{a}College of Computer and Information, Hohai University, PR China
\textsuperscript{b}State Key Laboratory of Novel Software Technology, Nanjing University, PR China
\textsuperscript{c}Department of Mathematics, Nanjing University, Nanjing, PR China
\textsuperscript{d}School of Information Science & Technology, East China Normal University, PR China

\textbf{A R T I C L E   I N F O}

\textbf{Article history:}
Received 26 January 2013
Received in revised form 3 June 2013
Accepted 15 June 2013
Available online 7 July 2013

\textbf{Keywords:}
Information retrieval
User profile
Semantic Web
Ontology
User ontology
Optimization

\textbf{A B S T R A C T}

User profiles play an important role in information retrieval systems. In this paper, we propose a novel method for the acquisition of ontology-based user profiles. In the method, the ontology-based user profiles can maintain the representations of personal interest. In addition, user ontologies can be automatically constructed. The method can make user profiles strong expressive and less manually interfered.

\copyright 2013 Elsevier Ltd. All rights reserved.

\section{1. Introduction}

The ability of facilitating users to achieve useful information is more and more important for information retrieval systems. User profiles can represent personal interests. These representations can be utilized for information retrieval. User profiles can be leveraged to recommend the truly valuable information to the users.

Generally, there are two major methods of collecting user profiles \cite{1}. Firstly, preprocessing is a method done with a questionnaire. This method is time consuming for users. Secondly, automatic collection is based on the interaction between the user and the system. When a user frequently uses a system, the cost of automatic collection is not more expensive than that of preprocessing \cite{1}. However, when a user rarely uses a system, it is difficult for the system to represent the user’s personal interests. To solve this problem, the user has to repeat a similar procedure for each system and therefore this increases user burden \cite{1}.

With emergence of Semantic Web, user ontology has more opportunities to improve user profiles. User ontology can share information between systems. Even if a user rarely uses a system, the system may share relevant information in other systems and represent the user’s personal interests. However, manual ontology construction is a time-consuming and expensive task. Thus, we propose a method called OUPA (Ontology-based User Profiles Acquisition) to acquire user-profiles through the construction of user ontologies. In the OUPA method, the ontology-based user profiles can maintain the representations of personal interest. In addition, user ontologies can be automatically constructed to avoid the problem of time consuming and expensive manual construction task using manual ontology building.

\section{2. Related work}

Navigli et al. \cite{2,3} present a system, called OntoLearn. The OntoLearn system employs a combination of inductive machine learning and natural language processing for the extraction of domain ontologies from a corpus of domain text. In the OntoLearn system, semantic interpretation is used to identify the concepts and the semantic relations among the concepts, and a method called SSI is used for sense classification. More recently, Wei et al. \cite{4} employ probabilistic topic models to learn relationships between a set of concepts. Particularly, they employ the LDA model to learn concept hierarchies.

Sánchez et al. \cite{5} propose a method of automatically acquiring non-taxonomic relationships for constructing domain ontologies. This method discovers and selects domain-related verb phrases as non-taxonomic labels, and labels the non-taxonomic relationship according to the retrieved verbs relating a pair of concepts. More recently, Villaverde et al. \cite{6} exploit the syntactic structures and dependencies among concepts existing to acquire non-taxonomic relationships and extract verbs that are frequently
applied for linking the related concepts. They employ a popular association rule mining algorithm to discover concepts and verbs that frequently co-occur from a domain-specific text corpus.

Wu and Weld [7] present an autonomous system called KOG (Kylin Ontology Generator) for refining the wikipedia infobox ontology. In KOG, support-vector machines (SVMs) and Markov Logic Networks (MLNs) is used to build a rich ontology that is combined Wikipedia infoboxes with WordNet. More recently, Lau et al. [8] propose a method of automatically extracting lightweight fuzzy ontology to create a formal fuzzy concept maps for representing learners' knowledge structures. This method uses the notion of fuzzy ontology to create a formal fuzzy domain ontology model.

Han et al. [9] present an approach to acquiring ontology information from Web documents. The method employs association rules and inference mechanism to acquire association instances of ontology classes and employs clustering to construct ontology. Han [10] presents a method of the creation of ontology-based adaptive user profiles. The method employs a combination of collaborative filtering, association rules, simulated annealing and fuzzy genetic algorithms to create user profiles and employs a combination of complete-link clustering algorithms and inference mechanism to create user ontologies. Han et al. [11] employ a combination of complete-link clustering and inference for the creation of semantic annotations and employ the k-nearest-neighbor query for the maintenance of semantic annotations. Han et al. [12] employ an augmented Lagrangian function to build fuzzy clustering model and employ the fuzzy c-means combined with simulated annealing and PHR to build ontology-based user profiles. Han and Chen [13] employ ontology-based collaborative filtering to discover semantic relationships that are acquired from the constructed ontologies and employ RBF neural networks to find the most relevant web documents and their corresponding terms from these similar users' queries. Han et al. [14] employ a combination of genetic algorithm, self-organizing map, association rules and inference mechanism for the creation of semantic annotations and employ a combination of k-nearest-neighbor query and simulated annealing for the maintenance of semantic annotations. Han and Yan [15] employ compromise programming combined with pattern search to construct biclusters and to identify user communities.

In contrast to the above related work, our OUPA method considers two components of constructing the user profiles based on ontologies. Firstly, we propose the algorithm which introduces simulated annealing to hierarchical clustering in the construction of user ontologies. Secondly, we propose the k-nearest-neighbor query method combined with branch-and-bound search algorithm in the maintenance of user ontology. The method has less human intervention in creating user profiles and can make user profiles more strong expressive.

3. OUPA method

The OUPA method is used to acquire hierarchically structured ontology-based user profiles. The OUPA method consists of the CUO (The Construction of User Ontologies) method and the MUO (The Maintenance of User Ontology) method. The CUO method is used to construct user ontologies, and the MUO method is used to maintain user ontology. The CUO method is described in Section 3.1, and the MUO method is described in Section 3.2.

In the OUPA method, the user profiles can be constructed from the Web resources which are some personal web pages related to their name, title, research interests, publications, research projects etc. In OUPA, a user profile corresponds to several user ontologies, the same ontology can be shared by several user profiles, and an ontology-based user profile is used to represent personal interests in order that a user may own several interests. For example, there are two user ontologies faculty and student. In ontology faculty, faculty is an entity superclass, and professor, associate professor and assistant professor are entity subclasses. Name, positions, and students in faculty are attributes. Ontology faculty is constructed. In the other ontology student, student is an entity superclass, and undergraduate student and graduate student are entity subclasses. At the next lower level, graduate student is an entity superclass, and Master student and Ph.D. student are entity subclasses. Name, advisor and research interests in graduate student are attributes. Ontology student is constructed. Once these entity classes are instanced, user profiles can be expressed. There exists the relationship between faculty and student, therefore Xun Zhang's user profile is described as follows: (Xun Zhang, professor, [(Lingxin Yun, information retrieval), (Guangan Fei, database), (Yun Wu, semantic Web)])

Continued on next page
is the squared error, $n_i$ is the number of entity in cluster $r$, $x_{ri}$ is entity $i$ in cluster $r$, and $x_r$ is the centroid of cluster $r$.
3. New solution is produced by randomly choosing another entity from neighboring region $N(x_r)$.
4. Objective function difference is $\Delta f = f(\text{new entity}) - f(\text{local optimal solution})$.
5. Accept criterion is $P = 1$ for $\Delta f > 0$ or $P = \exp(-\Delta f/t)$ for $\Delta f < 0$.

The ISAC algorithm is described as follows:
**Input:** a set of entities, initial solution $x_0$, function $f$, initial temperature $t_0$, initial step $q_0$, monotone increment function $I(x)$

**Output:** optimal solution

\[
\begin{align*}
\{ & x' = x_0; // initialize optimal solution x' \\
& f' = f_0; // initialize optimal objective function value f' \\
& q = q_0; // initialize step q \\
& t = t_0; // initialize temperature t=
\end{align*}
\]

\[i = 1; // initialization\]
\[at = 1; // initialization\]
\[\text{while } |R_1 - C| > \varepsilon // is a given enough small positive number, where $\varepsilon = 9 \times 10^{-2}$. Threshold C is 0.82\]
\[|R_{i-1} = at(as); // at is the acceptable times of new solution, as is iterative steps number, $R_{i-1}$ is acceptable rate \]
\[at = at + 1; \]
\[as = 0; // initialization\]
\[\text{while } f < U // U is the upper limit of step length \]
\[\{ \text{as = as + 1;} \]
\[\text{while } x \notin N(x) // N(x) is neighboring region \]
\[\{ \text{step length is q and the BFGS optimizer in the quasi-Newton method is used to compute local optimal solution;} \]
\[\text{if } x_i \text{ is produced by randomly choosing another entity from neighboring region } N(x) \text{ and objective function difference } \Delta f = f(x_i) - f(x) \text{ is computed;} \]
\[\text{if } \Delta f \leq 0 \text{ or } \exp(-\Delta f/t) > \text{random}(0,1) \]
\[\{ \text{then } (x = x_i); \]
\[\text{if } f < f' \]
\[\{ \text{then (} x = x'; \]
\[f = f'; \]
\[\} \]
\[\text{if } at \text{ is equal to the given acceptable times then break;} \]
\[\} \]
\[q = I(R_1 \times q); // adjustment function I(x) is monofonic increasing function, I(x) = (x - 0.5)^2 + 1\]
\[\text{if } f < f' \]
\[\{ \text{then } (x = x'; \]
\[f = f'; \]
\[\} \]
\[R_i = at(as); \]
\[\text{if } i > 2 \]
\[\text{then (compute self-adjusting temperature}\]
\[\{ \text{if } R_{i-1} < C \text{ and } R_i < C \]
\[\text{then } t = t + TC; // TC is a given constant}\]
\[\text{if } R_{i-1} \geq C \text{ and } R_i \geq C \]
\[\text{then } t = t - TC; \]
\[\text{if } R_{i-1} \leq C \text{ and } R_i \geq C \]
\[\text{then } t = t + TC/2; \]
\[\text{if } R_{i-1} \geq C \text{ and } R_i < C \]
\[\text{then } t = t + TC/2; \]
\[\} \]
\[i = i + 1; \]
\} \]

3.1.2. ACUO algorithm

Based on the ISAC algorithm, the ACUO algorithm is used to automatically construct user ontologies. The ACUO algorithm is a divisive clustering algorithm for hierarchical clustering [21–23]. In a divisive clustering algorithm, all entities are first considered to be in one cluster. The farthest two entities in the same cluster are used as seed points for two clusters. All entities that have the closest seed in the old cluster are placed into the new cluster. This procedure continues until smaller subsets are produced until a threshold distance is reached. As described in the ISAC algorithm, the ACUO algorithm introduces simulated annealing to analyze the semantic relationships between entities, and construct the hierarchically structured user ontologies. Accordingly, the construction of user ontologies is converted into the global optimization problem for clustering. The time complexity of the ACUO algorithm is $O(2^l - 1) \times (\text{the time complexity of the ISAC algorithm})$, where $l$ is the number of layers in user ontology.

The ACUO algorithm is described as follows:

**Input:** the entire entities set

**Output:** user ontologies

\{(all entities in the entire dataset are regarded as one cluster; \}
\{while one cluster does not consist of only a single entity \}
\{(the ISAC algorithm is used to compute optimal solution, \}
\{so that one cluster is broken down into two subsets; \}
\{every subset is regarded as one cluster; \}
\}\}

In contrast to other divisive clustering algorithms such as Principal Direction Divisive Partitioning (PDDP) algorithm for hierarchical clustering [22] and the non-greedy version of the PDDP (NGPDDP) algorithm [23], the ACUO algorithm introduces simulated annealing to clustering for the construction of user ontologies. A simulated annealing algorithm ISAC can break down the data into sub-clusters more reasonably.

3.2. The maintenance of user ontology

The maintenance of user ontology aims to update some parts of the ontology, in order to perform incremental clustering for supporting insertions and deletions after initial clusters are formed. The MUO method consists of the BBU (Introduce Branch and Bound for Updating) algorithm and the AMUO (Automating the Maintenance of User Ontology) algorithm. The BBU algorithm is used to evaluate the nodes close enough to the entity which will be inserted or deleted. Based on the BBU algorithm, the AMUO algorithm is used to maintain user ontology. The BBU algorithm is described in Section 3.2.1, and the AMUO algorithm is described in Section 3.2.2.

3.2.1. BBU algorithm

We propose an efficient branch-and-bound search algorithm called BBU to evaluate the nodes close enough to the entity which will be inserted into a user ontology or deleted from a user ontology. The BBU algorithm introduces a branch and bound method [24,25] to evaluate the entity classes of user ontology close enough to the specified entity which is inserted or deleted. Specifically, the BBU algorithm examines the connectivity of entity classes in user ontology, traverses the semantic relationships between entity classes before a given threshold is reached, and finds the closest entity classes according to their semantic distance.

In the following description, $k$ stands for the iteration index, $List_k$ denotes the list in which the candidate class nodes are listed, $U_k$ denotes the upper bound at the end of $k$ iterations, and $x'$ denotes optimal solution. The bounding function $F(x) = \max(\cos(d_1, d_2))$, where $d_1$ and $d_2$ are two classes vectors. $\cos(d_1, d_2) = (d_1 \cdot d_2)/
3.2.2. AMUO algorithm

We propose the AMUO algorithm which introduces a \( k \)-nearest-neighbor query method and branch-and-bound search algorithm BBU to maintain a hierarchically structured user ontology. The time complexity of the AMUO algorithm is \( O(l + n \times \log_2 n) \), where \( l \) is the number of layers in user ontology and \( n \) is the number of entity classes.

The AMUO algorithm is described as follows:

**Input:** user ontology, a specified entity from a user ontology. The time complexity of the BBU algorithm is \( O(n \times \log_2 n) \), where \( n \) is the number of entity classes.

**Output:** optimal solution \( x' \)

\( x' = \{ \text{the children of the root node is added to a queue; } \}

\( x' = \{ \text{the children of this node is added to a queue; } \}

\( x' = \{ \text{the entity is inserted into the leaf node; } \}

\( x' = \{ \text{the entity is deleted from the leaf node; } \}

\( x' = \{ \text{the subtree represented is discarded by this node; } \}

then if the node and any of its descendants are not close enough to the entity
then the children of this node is added to a queue;
else (the entity is deleted from the leaf node;
if there are "too few" entity classes in the leaf node
then the neighboring leaf nodes are merged;

The AMUO algorithm is different from Roussopoulos et al. [26] in that the AMUO algorithm is used to maintain a hierarchically structured user ontology, while Roussopoulos et al. [26] is used to process exactly the \( k \) nearest neighbors of a given query point in space for the R-trees. In contrast to Boley [22] and Cutting et al. [27] and Li et al. [28], the AMUO algorithm is a hybrid algorithm of both \( k \)-nearest-neighbor query method and branch-and-bound search algorithm for maintaining user ontology, where a branch-and-bound search algorithm BBU is used to improve search efficiency.

4. Experimental results and discussion

In this section, a case study has been given and some experiments have been done to evaluate the performance of the proposed method. In the case study, six user ontologies are described below: user ontology 1 is position, user ontology 2 is organization, user ontology 3 is teaching, user ontology 4 is project, user ontology 5 is publication and user ontology 6 is educational background. There are nine entities in user ontology 1. Nine entities in user ontology 1 include Research Fellow, Senior Research Associate, Senior Research Assistant, Professor, Associate Professor, Assistant Professor, Lecturer, Senior Lecturer and Reader. The construction process of user ontology 1 is shown in Fig. 1. Number 1–9 in Fig. 1 respectively denote Research Fellow, Senior Research Associate, Senior Research Assistant, Professor, Associate Professor, Assistant Professor, Lecturer, Senior Lecturer and Reader. There are seven entities in user ontology 2. Seven entities in user ontology 2 include Name of Organization, Nature of Organization, Address of Organization, Post Code of Organization, Web Site of Organization, Fax of Organization and Name of Employer. The construction
process of user ontology 2 is shown in Fig. 2. Number 1–7 in Fig. 2 respectively denote Name of Organization, Nature of Organization, Address of Organization, Post Code of Organization, Web Site of Organization, Fax of Organization and Name of Employer. There are seven entities in user ontology 3. Seven entities in user ontology 3 include Name of Book, Teacher, Number of Student, Class of Student, Period of Course, Code of Course and Name of Course. The construction process of user ontology 3 is shown in Fig. 3. Number 1–7 in Fig. 3 respectively denote Name of Book, Teacher, Number of Student, Class of Student, Period of Course, Code of Course and Name of Course. There are five entities in user ontology 4. Five entities in user ontology 4 include Category of Project, Number of Project, Name of Project, Period of Project and Sum of Project. The construction process of user ontology 4 is shown in Fig. 4. Number 1–5 in Fig. 4 respectively denote Category of Project, Number of Project, Name of Project, Period of Project and Sum of Project. There are seven entities in user ontology 5. Seven entities in user ontology 5 include Title, Author, Source of Index, Name of Publication, Data of Publication, Volume of Publication and Issue of Publication. The construction process of user ontology 5 is shown in Fig. 5. Number 1–7 in Fig. 5 respectively denote Title, Author, Source of Index, Name of Publication, Data of Publication, Volume of Publication and Issue of Publication. There are six entities in user ontology 6. Six entities in user ontology 6 include Student Level, Primary Students, Students of High School, Undergraduates, Master’s Degree Students and Doctoral Degree Students. The construction process of user ontology 6 is shown in Fig. 6. Number 1–6 in Fig. 6 respectively denote Student Level, Primary Students, Students of High School, Undergraduates, Master’s Degree Students and Doctoral Degree Students.

Experiments are conducted to evaluate the precision of the proposed method. The concept of precision is used to evaluate the experimental results of the method. The precision is the proportion of entities that is exactly constructed in the hierarchically structured user ontologies. Six user ontologies are constructed to show that the proposed method has a high precision. Six user ontologies are described below: user ontology 1 is position, user ontology 2 is organization, user ontology 3 is teaching, user ontology 4 is project, user ontology 5 is publication and user ontology 6 is educational background. The precisions of these user ontologies are shown respectively in Table 1. The experiment result in Table 1 shows that
The precision of user ontology.

<table>
<thead>
<tr>
<th>user ontology</th>
<th>precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>85.6%</td>
</tr>
<tr>
<td>2</td>
<td>86.0%</td>
</tr>
<tr>
<td>3</td>
<td>84.0%</td>
</tr>
<tr>
<td>4</td>
<td>80.0%</td>
</tr>
<tr>
<td>5</td>
<td>78.0%</td>
</tr>
<tr>
<td>6</td>
<td>74.0%</td>
</tr>
</tbody>
</table>

5. Conclusion

User profiles have an important role in information retrieval system. With emergence of Semantic Web, user ontology provides opportunities for improving user profiles. User ontology can maintain the representations of personal interest. In this paper, a method called OUPA is proposed for the acquisition of ontology-based user profiles. One key feature of OUPA is that we introduce simulated annealing to cluster partition for automatically constructing user ontologies. Another key feature of OUPA is that it is a hybrid algorithm of both k-nearest-neighbor query method and branch-and-bound search algorithm is proposed for maintaining user ontology.

Acknowledgements

The authors thank Professor Linping Sun for technical help. The authors thank the anonymous reviewers for their constructive comments. This work is supported by the National Natural Science Foundation of China (Project 60571048, Project 60873264, Project 60971088, Project 61272402, and Project 61070214), and the Qing Lan Project.

References